AFRL-AFOSR-UK-TR-2015-0014

# Consistent evolution of software artifacts and non-functional models

**Professor Vittorio Cortellessa**

**UNIVERSITA DEGLI STUDI DELL AQUILA**
**VIA VETOIO 1**
**LAQUILA, , 67100, ITALY**

**EOARD GRANT #FA8655-11-1-3055**

Report Date: November 2014

Final Report from 1 October 2011 to 30 September 2014

**Air Force Research Laboratory**
**Air Force Office of Scientific Research**
**European Office of Aerospace Research and Development**
**Unit 4515, APO AE 09421-4515**

| REPORT DOCUMENTATION PAGE | | Form Approved OMB No. 0704-0188 |
|---|---|---|

| | | |
|---|---|---|
| **1. REPORT DATE** ***(DD-MM-YYYY)*** 14 November 2014 | **2. REPORT TYPE** Final Report | **3. DATES COVERED** *(From – To)* 1 October 2011 – 30 September 2014 |

| | |
|---|---|
| **4. TITLE AND SUBTITLE** **Consistent evolution of software artifacts and non-functional models** | **5a. CONTRACT NUMBER** |
| | **5b. GRANT NUMBER** **FA8655-11-1-3055** |
| | **5c. PROGRAM ELEMENT NUMBER** 61102F |
| **6. AUTHOR(S)** **Professor Vittorio Cortellessa** | **5d. PROJECT NUMBER** |
| | **5d. TASK NUMBER** |
| | **5e. WORK UNIT NUMBER** |
| **7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)** UNIVERSITA' DEGLI STUDI DELL'AQUILA VIA VETOIO 1 L'AQUILA, 67100, ITALY | **8. PERFORMING ORGANIZATION REPORT NUMBER** N/A |
| **9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)** EOARD Unit 4515 APO AE 09421-4515 | **10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL/AFOSR/IOE (EOARD) |
| | **11. SPONSOR/MONITOR'S REPORT NUMBER(S)** **AFRL-AFOSR-UK-TR-2015-0014** |

**12. DISTRIBUTION/AVAILABILITY STATEMENT**

**Distribution A: Approved for public release; distribution is unlimited.**

**13. SUPPLEMENTARY NOTES**

**14. ABSTRACT**

This project has focused on the problem of filling the gap between non-functional analysis of software models and software development artifacts. Its main motivation stems from the fact that, still today, non-functional analysis and software development activities are quite loosely interconnected, thus the additional value that they can lead to each other is often lost. Instead it is commonly recognized that a tighter interconnection of practices, instruments and notations would give rise to more accurate analyses and more effective feedback from analysis results back to software artifacts. The confidence that advanced model-driven techniques can be suitable within this context came from previously successful collaborations among members of our research group that includes researchers from the Model-Driven Engineering and from the Non-functional Analysis areas. The seed of this project's activities was our first experiments on bidirectional transformations between software artifacts and performance models. Our intent was to understand whether this type of transformations could close the loop in both directions that are: transforming software artifacts into analyzable performance models (forward path), interpreting performance analysis results in order to remove problems from performance models and propagate the changes back to software artifacts (backward path). Starting from that point, the project has addressed three main research questions that, along the project development, have emerged as key points in the backward path context, which is the most problematic one. They can be summarized as follows: (1) Is it more advantageous working on the performance side or on the software side to solve problems identified with performance analysis? Can synergy be found among techniques adopted on these sides? (2) How can advanced model-driven techniques help in the model refactoring that is required to remove performance problems detected from the analysis? and (3) How performance problem detection can be driven towards the most critical problems (i.e. the ones that more heavily induce bad software performance)?

**15. SUBJECT TERMS**

EOARD, Nano particles, Photo-Acoustic Sensors, Model-Driven Engineering (MDE), Software Performance Engineering (SPE), Change Propagation, Performance Antipatterns

| **16. SECURITY CLASSIFICATION OF:** | | | **17. LIMITATION OF ABSTRACT** | **18, NUMBER OF PAGES** | **19a. NAME OF RESPONSIBLE PERSON** James H Lawton, PhD |
|---|---|---|---|---|---|
| **a. REPORT** UNCLAS | **b. ABSTRACT** UNCLAS | **c. THIS PAGE** UNCLAS | SAR | 6 | **19b. TELEPHONE NUMBER** *(Include area code)* (703) 696-5999 |

# Consistent evolution of software artifacts and non-functional models
# EOARD Grant/Cooperative Award no. FA8655-11-1-3055
# Final Report : 1 October 2011 - 30 September 2014

Cortellessa V., Di Marco A., Di Ruscio D., Pierantonio A.,
Arcelli D., Eramo R., Trubiani C., Tucci M.

*Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica,*
*Università degli Studi dell'Aquila,*
*Via Vetoio, 67100 L'Aquila, Italy*
*Email: vittorio.cortellessa@univaq.it*
*Web: http://www.di.univaq.it/cortelle/*
*Phone: +390862433165 , Fax: +390862433131*

---

**Abstract**

This project has tackled the problem of keeping aligned software artifacts and non-functional models along the software lifecycle, with a specific focus on software performance aspects. For this goal, we have experimented model-driven techniques that allowed to introduce automation in the (forward and backward) propagation of software model changes. This document reports the project results.

---

**Keywords:** Model-Driven Engineering (MDE), Software Performance Engineering (SPE), Change Propagation, Performance Antipatterns.

*For sake of readability of the project results, we have summarized in this document the research questions addressed and the results obtained, whereas for details about methodologies and techniques adopted to achieve these results the readers can refer to the 10 papers published (plus 1 paper under revision) thanks to this project funding and listed in the bibliography.*

## 1. Introduction

This 3-years project has focused on the problem of filling the gap between non-functional analysis of software models and software development artifacts. Its main motivation stems from the fact that, still today, non-functional analysis and software development activities are quite loosely interconnected, thus the additional value that they can lead to each other is often lost. Instead it is commonly recognized that a tighter interconnection of practices, instruments and notations would give rise to more accurate analyses and more effective feedback from analysis results back to software artifacts.

The confidence that advanced model-driven techniques can be suitable within this context came from previously successful collaborations among members of our research group that includes researchers from the Model-Driven Engineering and from the Non-functional Analysis areas.

The seed of this project activities was our first experiments on bidirectional transformations between software artifacts and performance models. Our intent was to understand whether this type of transformations could close the loop in both directions, that are: transforming software artifacts into analyzable performance models (forward path), interpreting performance analysis results in order to remove problems from performance models and propagate the changes back to software artifacts (backward path) [1].

Starting from that point, the project has addressed three main research questions that, along the project development, have emerged as key points in the backward path context, that is the most problematic one [2]. They can be summarized as follows:

RQ1 Is it more advantageous working on the performance side or on the software side to solve problems identified with performance analysis? Can synergy be found among techniques adopted on these sides (e.g. bottleneck analysis on performance side and antipattern detection and removal on software side)?

RQ2 How can advanced model-driven techniques help in the model refactoring that is required to remove performance problems detected from the analysis?

RQ3 How performance problem detection can be driven towards the most critical problems (i.e. the ones that more heavily induce bad software performance)?

Several results have been obtained for all the above research questions. The remainder of this report is organized as follows: the next three sections summarize the results obtained for each question, then conclusions and research perspectives are sketched.

## 2. Software or performance side? (RQ1)

Several approaches have been introduced in the last few years to tackle the problem of interpreting model-based performance analysis results and translating them into architectural feedback. Typically the interpretation can take place by browsing either the software model or the performance model. In [3] we have compared two approaches that we have introduced for this goal: one based on the detection and solution of performance antipatterns, and another one based on bidirectional model transformations between software and performance models. We have applied both approaches to the same example in order to illustrate the differences in the obtained performance results. Thereafter, we have raised the level of abstraction and we have discussed pros and cons of working on the software side and on the performance side.

Then we have investigated whether any synergy could be found among the techniques that are commonly adopted on the two sides, i.e. antipattern detection and solution on the software side and bottleneck analysis on the performance side. In [4], we aimed at showing that the approach combination allows to provide software engineers with broader sets of alternative solutions leading to better performance results. We have explored this research direction in the context of Layered Queueing Network models. After comparing the results achievable with each approach separately, we have quantitatively shown the benefits of merging bottleneck analysis and performance antipatterns.

## 3. Model-driven software refactoring (RQ2)

The contributions brought to this research question have been summarized in [5]. The target of our work in this direction has been to use metamodeling and modeling techniques to formalize refactoring actions aimed at removing detected performance problems.

In [6], we have introduced an approach that allows the refactoring of architectural models, based on antipatterns, aimed at providing performance improvement. To this end, we used a Role-Based Modeling Language to represent: (i) antipattern problems as Source Role Models (SRMs), and (ii) antipattern solutions as Target Role Models (TRMs). Hence, SRM-TRM pairs represent new instruments in the hands of developers to achieve architectural model refactorings aimed at removing sources of performance problems. Model refactoring for antipattern removal can be in fact obtained by replacing an SRM with the corresponding TRM. This approach has been applied to a case study, whose experimental results demonstrate its effectiveness.

Thereafter, in [7] we have represented each SRM-TRM pair as a difference model that encodes refactoring actions to be operated on a software model to remove the corresponding antipattern. Differences are applied to software models through a model transformation automatically generated by a higher-order transformation.

## 4. Identifying critical performance problems (RQ3)

The effectiveness of performance antipatterns detection is affected, like any pattern detection task, from the setting accuracy of thresholds that reveal the antipattern occurrence. For example, an antipattern occurs when too many messages are exchanged between two software components. But how many messages are too many? Hence, we have first investigated the influence of thresholds on the detection task. In [8] we have analyzed how a set of detected antipatterns may change while varying the threshold values. This work has been then consolidated in [9], where we have also studied the influence of thresholds on the complexity of refactoring actions, and we have quantified such influence using precision and recall metrics.

With reliable detection and refactoring techniques in the hands, the next problem we have tackled has been the introduction, in a performance-driven software evolution process, of metrics that can drive the process to convergence. In fact, when several performance antipatterns are detected, a critical issue is to decide how many and which ones to remove first. Starting from this point, we have developed two techniques: (i) one based on a metric of guilt that helps to identify the antipatterns that more heavily contribute to the violation of performance requirements [10], and (ii) another one aimed at working in a non-deterministic setting of thresholds, with the goal of identifying the antipatterns that more likely have occurred and more largely can improve the software performance once removed [11].

The former technique has been described in [10], where we have proposed an approach that provides software performance feedback to software engineers, since it suggests the design alternatives that allow overcoming the detected performance problems. The feedback process may be quite complex since engineers may have to assess several design options before achieving the architectural model that best fits the end-user expectations. In order to optimise such process we have introduced a ranking methodology that identifies, among a set of detected antipatterns, the guilty ones, i.e. the antipatterns that more likely contribute to the violation of specific performance requirements. The introduction of our ranking process leads the system to converge towards the desired performance improvement by discarding a consistent part of design alternatives.

The latter technique is described in [11], where we have worked in a fuzzy context with threshold values that cannot be determined, but only their lower and upper bounds do. On this basis, the detection task produces a list of performance antipatterns along with their probabilities to occur in the model. Several refactoring alternatives can be available to remove each performance antipattern. Our approach associates an estimate of how effective each alternative can be in terms of performance benefits. We have demonstrated that the joint analysis of antipattern probability and refactoring benefits drives the designers to identify the alternatives that heavily improve the software performance.

## 5. Conclusions

This 3-years project has allowed us to investigate multiple aspects of software evloution driven by performance aspects. As summarized in this report, the results of this project have been published in several conference and journal papers.

This work can be developed in future in several directions, where the following ones seem to us very promising.

First of all, it would be very interesting to experiment the techniques developed here on real-world examples of performance-critical systems. This would allow to consolidate the validation of the whole process. At the same time, the application of such techniques on specific application domains (such as Wireless Sensor Networks, Cloud, Safety-Crictical Systems, Adaptive Software Systems) would help to embed the process in specific context where perhaps some steps have to be refined in order to be more effective. For example, we have recently started to study the application of Feedback Control Theory to Adaptive Queueing Networks, that are performance models where several aspects can change in order to maintain a certain non-functional objective.

In another direction, the extension of this project results to other non-functional aspects of software would also be very interesting to investigate. Beside the canonical -ilities, such as reliability and availability, techniques that analyze tradeoff among these properties in order to generate feedback to software engineers are still lacking. This problem is ever more critical today, due to the increasing heterogeneity of environments where software is deployed, and also due to the adverse conditions under which software-based devices are required to be used (e.g. thermal conditions of sensors in different environments like ships, wind devices, data centers, etc.).

## References

[1] V. Cortellessa, R. Eramo, A. Pierantonio, M. Tucci, Performance-driven architectural refactoring through bidirectional model transformations, in: 8th International ACM Sigsoft Conference on the Quality of Software Architectures (QoSA 2012), 2012.

[2] V. Cortellessa, Performance antipatterns: State-of-art and future perspectives, in: Computer Performance Engineering - 10th European Workshop, EPEW 2013, Venice, Italy, September 16-17, 2013. Proceedings, volume 8168 of *Lecture Notes in Computer Science*, Springer, 2013, pp. 1–6.

[3] D. Arcelli, V. Cortellessa, Software model refactoring based on performance analysis: better working on software or performance side?, in: B. Buhnova, L. Happe, J. Kofron (Eds.), FESCA, volume 108 of *EPTCS*, 2013, pp. 33–47.

[4] C. Trubiani, A. D. Marco, V. Cortellessa, N. Mani, D. C. Petriu, Exploring synergies between bottleneck analysis and performance antipatterns, in: K. Lange, J. Murphy, W. Binder, J. Merseguer (Eds.), ACM/SPEC International Conference on Performance Engineering, ICPE'14, Dublin, Ireland, March 22-26, 2014, ACM, 2014, pp. 75–86. URL: http://doi.acm.org/10.1145/2568088.2568092. doi:10.1145/2568088.2568092.

[5] D. Arcelli, Model-based software refactoring driven by performance analysis, in: Doctoral Symposium at MODELS 2014, 2014.

[6] D. Arcelli, V. Cortellessa, C. Trubiani, Antipattern-based model refactoring for software performance improvement, in: 8th International ACM Sigsoft Conference on the Quality of Software Architectures (QoSA 2012), 2012.

[7] D. Arcelli, V. Cortellessa, D. Di Ruscio, Applying model differences to automate performance-driven refactoring of software models, in: Computer Performance Engineering - 10th European Workshop, EPEW 2013, Venice, Italy, September 16-17, 2013. Proceedings, volume 8168 of *Lecture Notes in Computer Science*, Springer, 2013, pp. 312–324.

[8] D. Arcelli, V. Cortellessa, C. Trubiani, Influence of numerical thresholds on model-based detection and refactoring of performance antipatterns, in: 1st Workshop on Patterns Promotion and Anti-patterns Prevention (co-located with CSMR 2013), 2013. URL: http://ppap.soccerlab.polymtl.ca/ppap2013/index.html.

[9] D. Arcelli, V. Cortellessa, C. Trubiani, Experimenting the influence of numerical thresholds on model-based detection and refactoring of performance antipatterns, ECEASST 59 (2013).

[10] C. Trubiani, A. Koziolek, V. Cortellessa, R. Reussner, Guilt-based handling of software performance antipatterns in palladio architectural models, Journal of Systems and Software 95 (2014) 141–165.

[11] D. Arcelli, V. Cortellessa, C. Trubiani, Performance-based software model refactoring in fuzzy contexts, in: Fundamental Approaches to Software Engineering, 2015. Under revision.